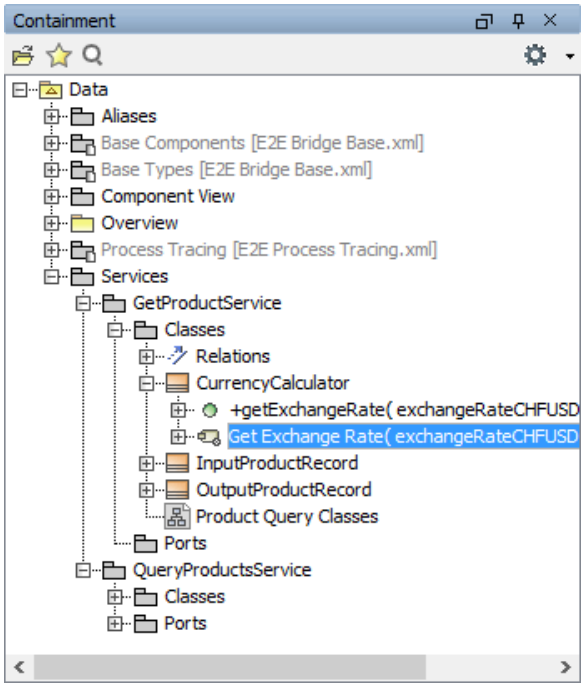
Implementing the Currency Calculator MD18

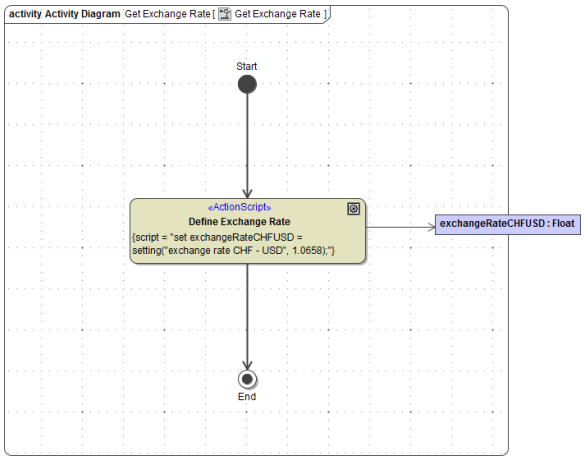
Before creating new activity diagrams, you should analyze some reuse options to reduce investments in time. The class **CurrencyCalculator** from the lesson 2 model can also be useful in lesson 3. You are going to copy it from lesson 2 and extend it afterwards.

Open the activity diagram **Get Exchange Rate** in package **Data / Services / GetProductService / Classes / CurrencyCalculator**.



In lesson 2, you created the activity diagram **Get Product** containing the call operation action **getExchangeRate** of class **CurrencyCalculator**.

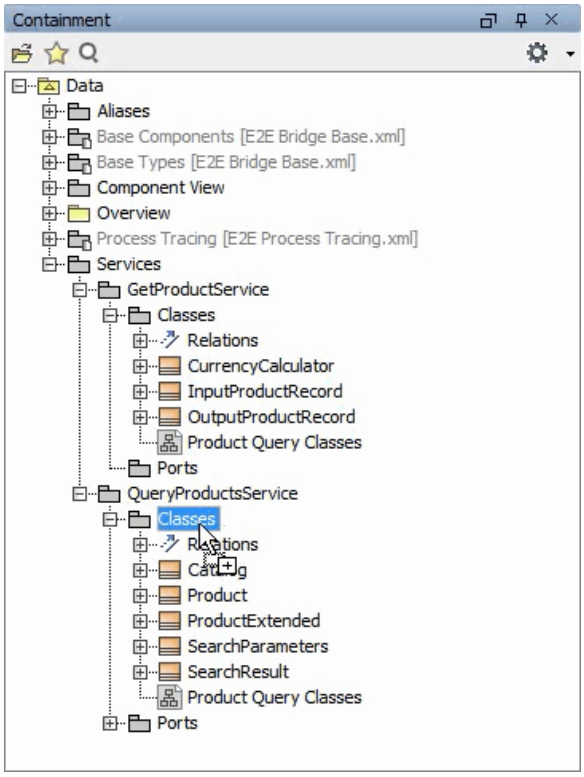
Navigate to the activity node **Get Exchange Rate** in the containment tree and double-click it to open the corresponding activity diagram.



In action node **Define Exchange Rate** in the activity diagram, the exchange rate is set to a fix value.

You will reuse the class operation **getExchangeRate** as you will perform currency conversions in this lesson, too. The hard coding will be substituted by a call of an external Web service.

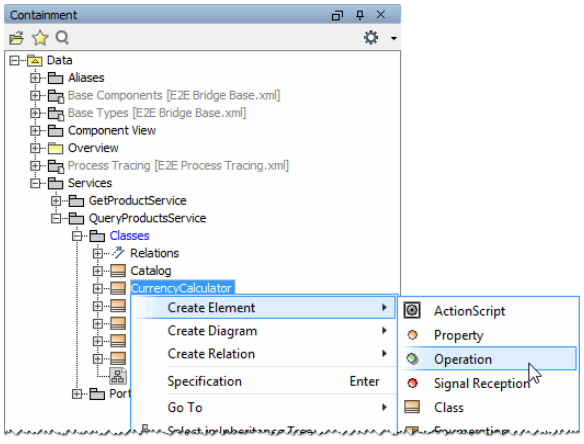
Getting the Exchange Rate from an External Web Service



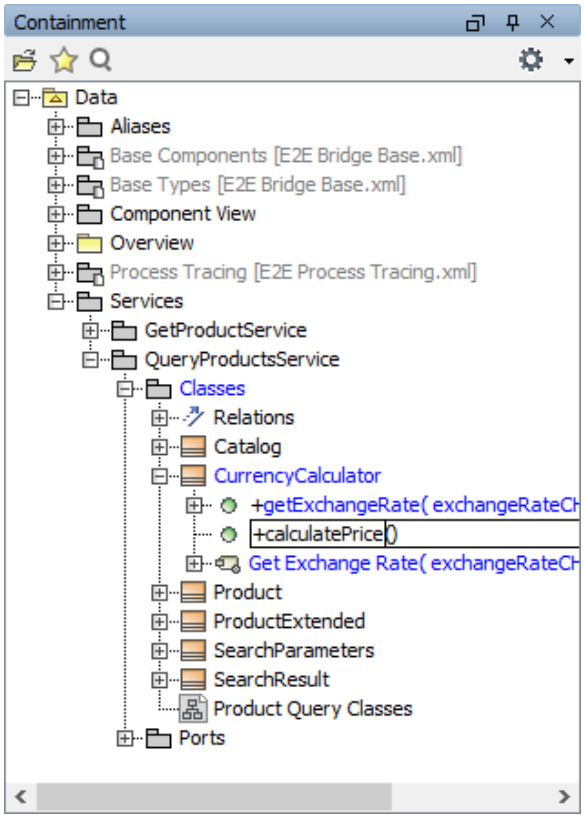
Drag and drop the whole class **CurrencyCalculator** from **GetProductService** to **QueryProductsService** to the package **Classes**.

Close the diagram **Get Exchange Rate**.

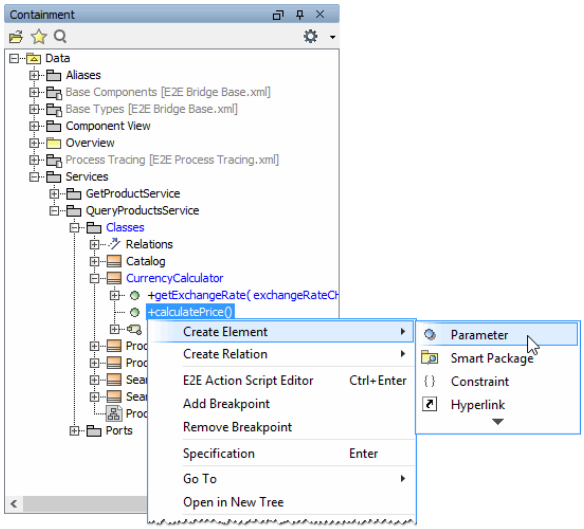
Now, you are going to extend the class **CurrencyCalculator** as to calculate a new price from a given price and exchange rate. This will be implemented in a new operation **calculatePrice**. For such small calculations, the E2E Model Compiler offers the possibility to insert action script directly into the operation without drawing a behavior diagram. The only restriction is that one parameter of direction **return** must be defined.



Right-click class **CurrencyCalculator** in the containment tree and select **Create Element > Operation**.



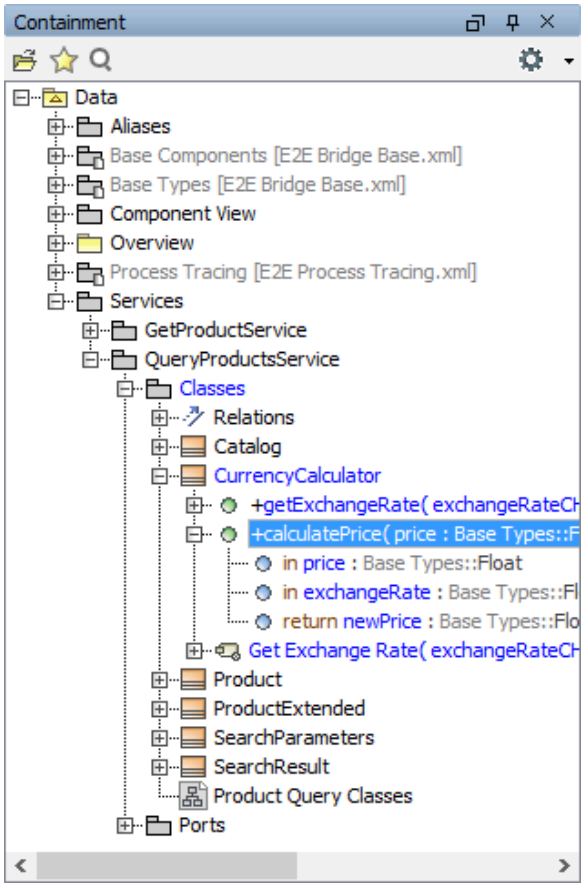
Assign the name **calculatePrice**.



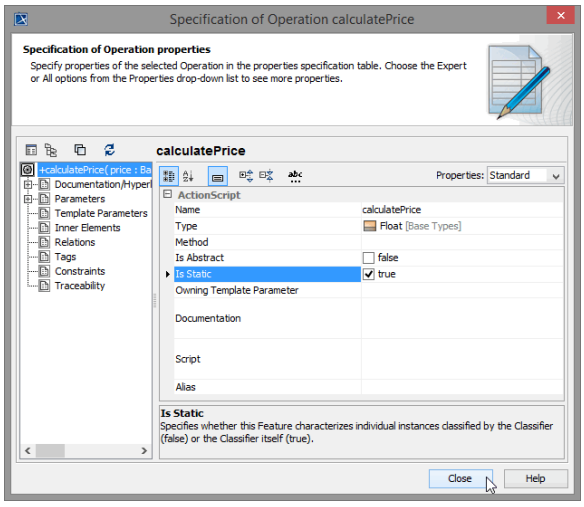
Define three parameters of the operation as listed below.

Pay attention to defining the direction of parameter **newPrice** as **return**.

Name	Type	Direction
price	Float	in
exchangeRate	Float	in
newPrice	Float	return

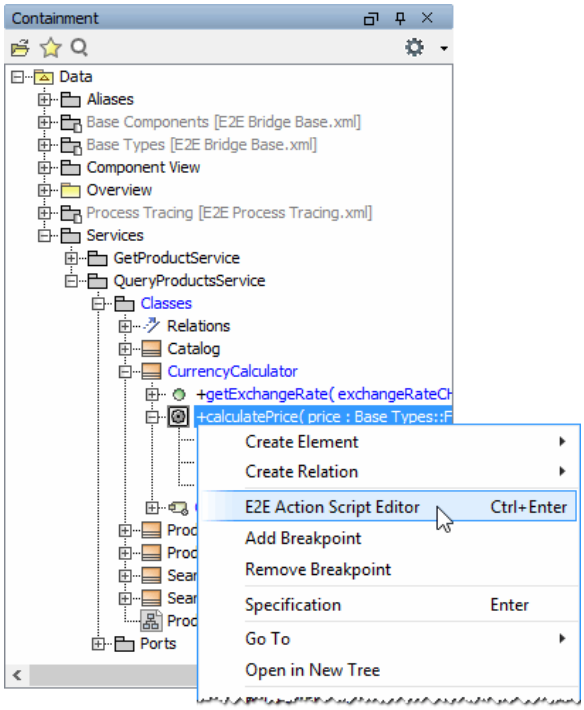


Double-click **calculatePrice** to open the **Specification** dialog.



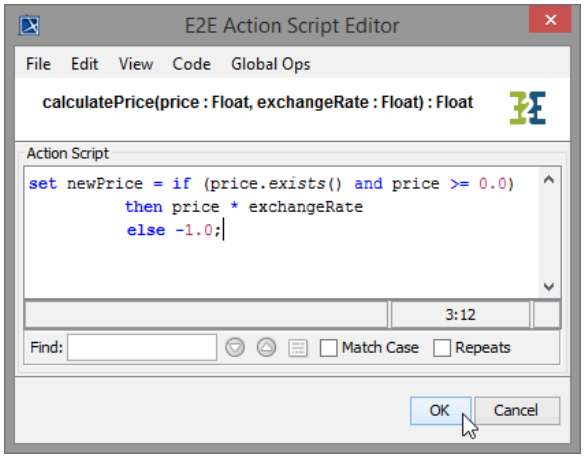
Apply the stereotype **Action Script** to the operation and define the operation as to be **static.**

Click **Close**.



Next to the operation name **calculatePrice** the icon is displayed to indicate the stereotype.

Select **E2E Action Script Editor** from the operation's context menu to directly add the action script to the operation.



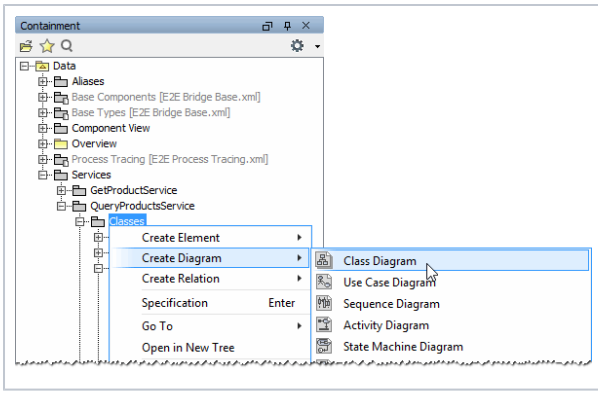
Enter the set-statement as shown on the left and click **OK**.

This statement is a combination of the `set`-statement with an if-clause. If parameter **price** exists and contains a valid value, the calculation `price * exchangeRate` is executed and the result is assigned to return parameter **newPrice**. In all other cases (`else`), **newPrice** is set to `-1.0`.

The newly defined static class operation **calculatePrice** will be used in the calculations part **Calculate Total and Currencies** of your model. You are going to call this operation directly from an action script without creating an object of type **CurrencyCalculator** first.

Therefore, you have to make it available to be used within action script. This is done via a <<use>> dependency from the port type **QueryProductsPortType** to the used class.

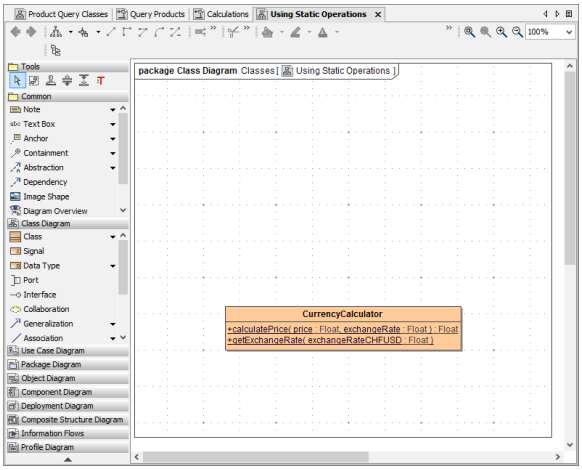
First, create a new class diagram in the package **Classes**.



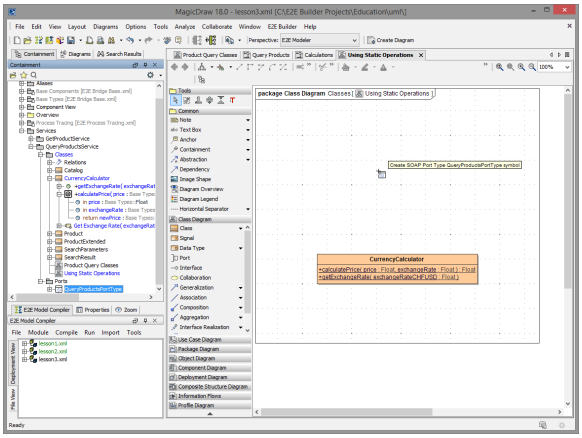
Select **Create Diagram > Class Diagram** from the context menu of the package **Classes**.

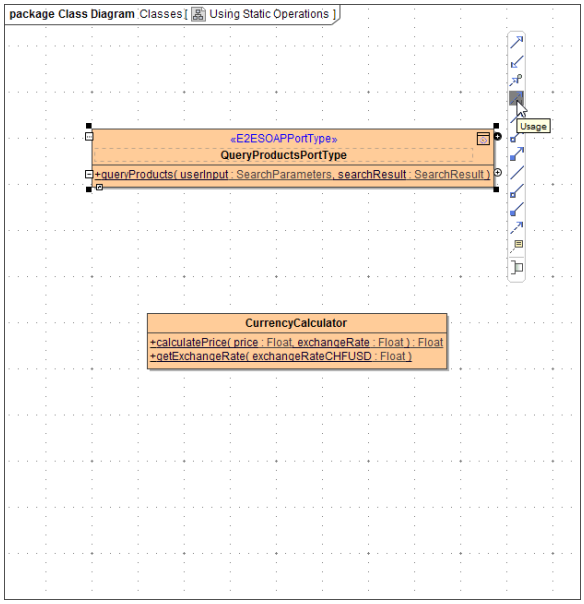
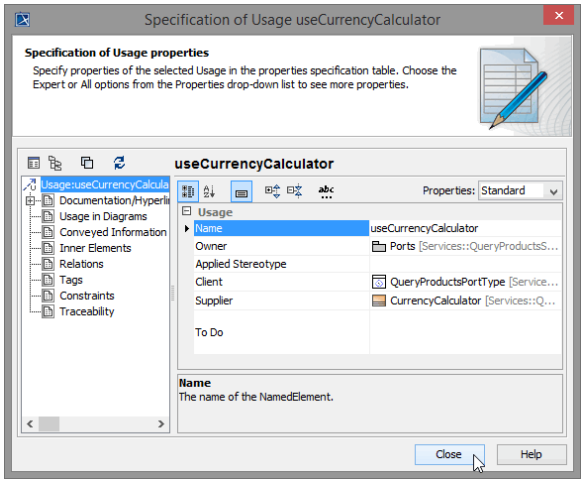
Assign the name **Using Static Operations**.

Drag the class **CurrencyCalculator** onto the diagram pane.

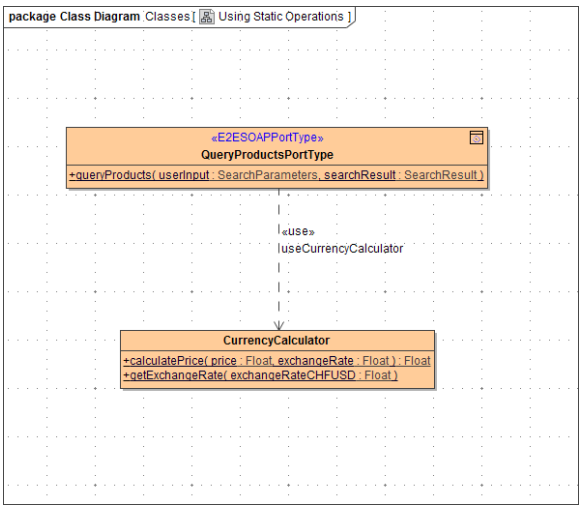


Drag the port type class **QueryProductsPortType** onto the diagram pane.



	Select the class symbol of **QueryProductsPortType** and - with help of the smart manipulation toolbar - draw a <<use>> dependency to class **CurrencyCalculator**.
	Open the **Specification** dialog of the <<use>> dependency and assign the name **useCurrencyCalculator**. Click **Close**.

Via the <<use>> dependency **useCurrencyCalculator** all static operations of **CurrencyCalculator** are now available in any action scripts that are part of **QueryProductsPortType**.



Save the UML model.